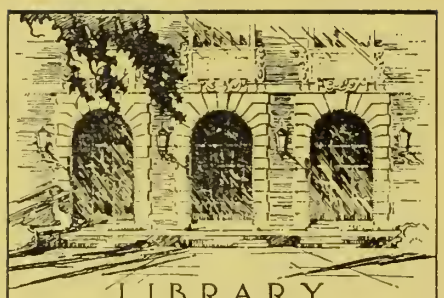


ILLIAC IV

QUARTERLY PROGRESS REPORT

October, November and December 1967

Contract No.
US AF 30(602)4144

ILLIAC IV Document No. 176

DEPARTMENT OF COMPUTER SCIENCE · UNIVERSITY OF ILLINOIS · URBANA, ILLINOIS

ILLIAC IV

QUARTERLY PROGRESS REPORT

October, November and December 1967

Contract No.
US AF 30(602)4144

Department of Computer Science
University of Illinois
Urbana, Illinois

February 1, 1968

This work was supported in part by the Department of Computer Science, University of Illinois, Urbana, Illinois, and in part by the Advanced Research Projects Agency as administered by the Rome Air Development Center, under Contract No. US AF 30(602)4144.

TABLE OF CONTENTS

1. Report Summary 1

2. Hardware 2

2.1 System Design. 2
2.1.1 Introduction. 2
2.1.2 Configuration Control. 2
2.1.3 Interrupt Control 2
2.1.4 Input/Output. 3
2.2 Diagnostic Programming. 3

3. Software 4

3.1 Language Translator Writing System. 4
3.1.1 Introduction. 4
3.1.2 Syntax Preprocessor 4
3.1.3 Symbol Scanner 4
3.1.4 Semantics and Intermediate Language 5
3.2 Tranquil 5
3.3 Gleipnir 6
3.4 System K 9
3.4.1 Assembler. 9
3.4.2 Simulator. 9
3.4.3 Debugging System 9
3.4.4 Operating System 9
3.4.5 B5500 Software Maintenance 10

4. Applications. 10

4.1 Partial Differential Equations 10
4.1.1 Introduction. 10
4.1.2 User Support Codes. 11
4.1.3 Slip Flow and Boundary Layer Studies. 13
4.1.4 Matrix Operations 15
4.1.5 Mathematical Service Routines 17
4.1.6 Carlson's S_n Method 17
4.2 Signal Processing 18
4.2.1 Application of ILLIAC IV to Phased Array Radar for Area Defense. 18
4.3 Matrices 18
4.3.1 Eigenvalue Problems 18
4.4 Linear Programming 19
4.4.1 Introduction. 19
4.4.2 LP Code 19
4.5 Computer Graphics 20
4.5.1 Introduction. 20
4.5.2 NCAR Model 20
4.5.3 Weather Mapping Algorithm 20
4.5.4 Graphical Display System. 21
4.6 Weapons Effects Calculations. 21
4.7 Anthropology Applications. 22
4.7.1 Population Genetic Program 22
4.7.2 Other Applications. 23

REFERENCES. 24

1. REPORT SUMMARY

Burroughs has executed its subcontract with Texas Instruments. The terms mirror precisely the terms in the contract between the University of Illinois and Burroughs.

The PE design is complete. However, implementation will be delayed by several months due to noise problems that have emerged with the 64 pin packages. It is hoped that the 64 pin packages (possibly expanded to 84 pins to permit providing a separate voltage supply to each gate group) will prove acceptable for the PE. It may be necessary to retreat to conventional IC's (14 or 16 pin dual inline packages) in the CU and PE memory. These considerations impact schedule and performance and have made it impossible to complete our rescheduling efforts during this quarter. It appears certain now, however, that during the next calendar quarter, successful progress can be made to predict accurately the impact on cost, schedule, and specifications.

The overall system design is now complete with the exception of the IOC. This report contains descriptions of configuration control, interrupt control, and the degree of synchronism between quadrants. The diagnostic programming effort has been initiated during this quarter, and it is expected that a preliminary detailed schedule of this task will be available at the end of the first quarter of 1969.

Work on the translator writer system and source language system continue on schedule. A unified approach to many problems in partial differential equations has been initiated and is outlined in this report. It is felt that this is a particularly important application effort, and future progress reports will contain summaries of progress as the work develops. In this effort, the help of many people in the community interested in large mesh calculations is being enlisted to achieve a useful definition of this system of programs.

2. HARDWARE

2.1 System Design

2.1.1 Introduction

During this period the multiprocessing capability of ILLIAC IV received attention for the system design activity. The activity includes the definition of configuration control, interrupt control, and I/O data transfer control.

2.1.2 Configuration Control

Three, 4-bit configuration control registers (CFC0, CFC1, CFC2) are employed for each CU. Each bit in a CFC corresponds to one CU. CFC0 defines a set of CU's which execute the same stream of instructions. CFC1 defines the array or arrays into which the instructions are loaded. CFC2 defines the array or arrays into which data for CU instructions are loaded.

CFC0 should specify one, two, or four CU's, one of them being its own CU. CFC1 and 2 can have any subset of the CU's specified by CFC0.

A concept of local and global CU instructions, which can be discriminated by a bit in the instruction, was adopted for simplicity. When the bit is zero (local instruction), the configuration control is bypassed and executed immediately in the CU; when the bit is one (global instruction), synchronization of the CU's specified by one of CFC's is requested before the execution of the instructions.

2.1.3 Interrupt Control

CU's handle interrupts individually in its own CU. This is due to a CU's non-synchronized operation, and most interrupts are local to each CU.

One of eight instruction buffer blocks is reserved for an interrupt handling routine; so as to hold the contents of the instruction buffer during the execution of the routine.

2.1.4 Input/Output

Software and hardware control over the whole I/O process is now being discussed. Some desired features have been identified and these are: 1) size of a data block and addressing and 2) memory for a queue of interrupts. Sixteen word data block and models of sixteen addressing registers have been suggested to handle small matrices and data descriptions. A memory for an interrupt queue is needed because one B6500 multiplier handles five different I/O's, namely DISK and four CU's. A part of the B6500 memory can be used for this purpose.

2.2 Diagnostic Programming

There are two efforts directed at developing algorithms for diagnostic programs. The first effort is to develop a computer program for checking data paths within ILLIAC IV. Data is input to the program in the form of a specification of the computer which includes a list of the component units and their connectivity and the constraints imposed by the machines' order code. For designated input-output combinations, the program will produce sequences of machine instructions that activate all possible paths from the input to the output. These sequences can then be used for fault testing. This program has been written and is in the process of checkout.

The second effort is to develop methods of checking LSI chips. The approach is to define input/output pins and to develop a minimal sequence of input patterns which will test all of the externally testable function on the chip. This method has been applied to the Carry-Save Adder (ILLIAC IV Doc. No. 160)[1], and a set of 107 test patterns for a PE exerciser unit has been determined. A set of 872 test patterns that can be implemented using the PE multiply operation has also been derived.

Burroughs has been informed of these efforts and will coordinate their fault diagnosis efforts with these.

3. SOFTWARE

3.1 Language Translator Writing System (TWS)

3.1.1 Introduction

The functional parts of the TWS were described in the last Quarterly Progress Report. The paragraphs below outline the progress in implementing the system.

3.1.2 Syntax Preprocessor

The syntax preprocessor which was designed and coded is being debugged. It scans the BNF productions in the syntactic metalanguage which specifies the syntax of a programming language (L) and generates a table of special terminal symbols, a table of nonterminal symbols, and a table which represents the productions. The first table will be used by the source program symbol scanner of the compiler for L to recognize either special terminal symbols or reserved words. The second table is, at the moment, discarded after a successful compiler has been generated. The third table is scanned by a procedure which attempts to convert it to Floyd Production Language form. Debugging is about to begin on this procedure, which is an implementation of the conversion algorithm described last quarter. Test of this algorithm on an unambiguous version of ALGOL 60 needed at most a two symbol lookahead. The output is a set of Floyd productions in the form of a sequence of pseudo orders which will be interpreted by a recognizer to parse a program in language L.

3.1.3 Symbol Scanner

The symbol scanner, which recognizes basic syntactic entities including identifiers, strings, literals, reserved words or special symbols, and single character terminals, is now operational. It performs table look up and entry on all but single character terminals. It converts literals, fixed or floating, with or without exponent and variable base which can range in value from 2 to 36, to a 64 bit ILLIAC IV word (the converted form is entered in the table).

A theoretical model has been created which reorders a linear list into a symmetric binary search tree. Its value is that it allows programs of just single character (or string) recognition capability to generate symmetric (structural) tables. The model has been designed to eliminate the need for reordering of data

3.1.4 Semantics and Intermediate Language

During this quarter, the language in which semantics is going to be specified to the TWS was defined and developed. As described in the previous report, the primary aim was to develop a language embodying the features of Burroughs extended ALGOL (used on the B5500 machine) and some special features to facilitate the description of the semantics of a programming language. Typical of these special features are commands for declaring and manipulating tables and stacks.

Some restrictions of the B5500 ALGOL compiler led to the conclusion that the only practical way of implementing the semantics language was by using a translator rather than direct usage of ALGOL procedures as originally planned. Thus, a language for describing the semantics of programming languages was tentatively defined (syntax and semantics). The language, called Illinois Semantic Language (ISL), is basically an extension of ALGOL; most of Burroughs extended ALGOL is allowed, and some additional commands were added to implement the special features mentioned above. The ISL Translator is an ALGOL program that accepts as input a program written in ISL and translates it into an ALGOL program. The translator is almost completely written, and debugging has been started.

3.2 Tranquil

This quarter marked the beginning of the implementation of the Tranquil compiler using the Translator Writing System.

The syntax specification for the declaration part of Tranquil has been redefined, and it now includes an option for the user to specify the maximum size of a dynamic array. The compiler then may determine an upper bound on likely memory requirements and hence determine the ILLIAC IV quadrant configuration best suited to the problem. The corresponding Pass I semantic routines are also being written.

In considering set declarations, a decision has been made to require the type of set involved to be declared along with other information if the set is dynamic (for storage reasons). This information will be placed in the main Pass I descriptor table; the fields of which have been specified for sets, as well as for arrays, variables, procedures, switches, and labels. Control statement syntax has been further specified, and productions have been linked with Pass I semantic routines for constructing the intermediate language to be analyzed during Pass II of the compiler.

The grammer for expressions was rewritten to the specifications required by the syntax analyzer. The Pass I semantic routines that correspond to these productions primarily involve transferring an operand from the input stack to the intermediate language output stream.

Because of the basically nonparallel nature of much of the work that must be done at the beginning of each block, a scheme has been devised to take partial advantage of ILLIAC IV parallelism by combining many strings or inherently sequential operations into a single, longer string of parallel operations. A program has been written to perform this conversion. This method is fairly good when the number of different operations involved is kept small and when the data bases of each of the different strings of operations are very similar. The criterion seems to be fairly well satisfied by the type of computations required at the beginning of a block.

3.3 Gleipnir

Preliminary specifications for a general purpose list processing language (Gleipnir) were completed in late September. This language is based on L^6[2] but has an ALGOL-like syntactic structure.

In order to improve its compatibility with Tranquil, a major revision of Gleipnir was completed in December. With this revision, Gleipnir retains the basic philosophy of L^6, but the treatment of variables and procedures is more like conventional ALGOL.

The basic entities of Gleipnir are base registers and field declarations. Base registers may be declared and used analogously to variables in ALGOL (with the exception of arrays). In addition, base registers may be used to hold pointers. Thus, complex data structures can be referenced via intial pointers contained in base registers.

Fields are declared in much the same way as base registers. These field definitions are applied to storage blocks (which are assigned dynamically by the storage allocator) as though one were applying a stencil to each block.

The following example is a procedure which searches a binary tree in each PE, and when a keyword is found, an associated 32 bit quantity is returned. Figure 1 shows how the data is arranged in one PE.

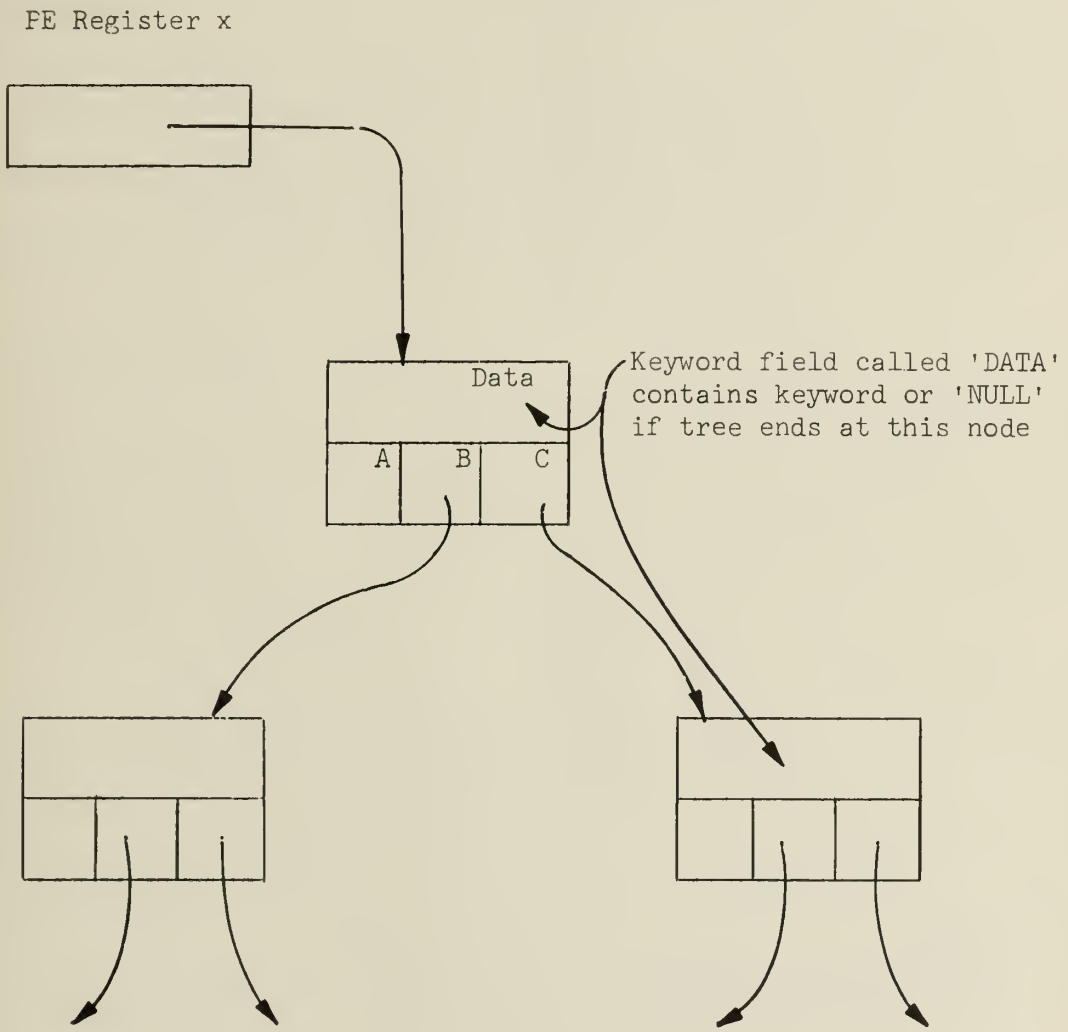


Figure 1. Arrangement of Data in One PEM for a Binary Tree

BINARY SEARCH

```
CU INTEGER PROCEDURE SEARCH (KEY, X); VALUE X;
          CU ALPHA REGISTER KEY;
          PE POINTER REGISTER X;
    BEGIN
          PE ALPHA FIELD DATA (0, 0, 63);
          PE ALPHA REGISTER WORD;
          PE POINTE FIELD B(1, 32, 47), C(1, 48, 63);
          PE INTEGER FIELD A(1, 0, 31);
          LABEL THERE, HERE;
          WORD ← KEY;
HERE:     FOR ALL X . DATA ≠ "NULL"  DO
    BEGIN
          FOR ONE X . DATA = WORD DO
               BEGIN
                 SEARCH ← X . A;
                 GO TO THERE
               END;
          FOR ALL X . DATA < WORD DO  X ← X . B;
          FOR ALL X . DATA > WORD DO  X → X . C;
    END;
    GO TO HERE;
THERE:
END PROCEDURE SEARCH;
```

3.4 System K

3.4.1 Assembler

During this quarter, the basic ILLIAC IV assembler was transferred from the 7094 to the B5500 by rewriting the assembler algorithm in Burroughs Extended ALGOL. This assembler has been exhaustively checked out and is available for use in conjunction with the ILLIAC IV simulator for basic checkout of ILLIAC IV code. (In this respect, see Document No. 159)[3].

3.4.2 Simulator

The simulator program has also been successfully transferred from the 7094 to the B5500. All orders of ILLIAC IV which are currently well defined have been implemented in the simulator and checked out. In association with the basic assembler, this program suite is now available for use by applications personnel.

3.4.3 Debugging System

In recognition of the novelty of some aspects of ILLIAC IV programming, it was decided to investigate the provision of more sophisticated debugging facilities than are generally provided on commercially available machines. These debugging facilities are to be ultimately formalized as a language extension to Tranquil. This work is currently in an early stage.

As a tooth-cutting project, an investigation is being made of the feasibility of attaching the debugging language extensions to B5500 ALGOL. A temporary grammer for a metastatement in ALGOL has been developed. Such a statement should allow programmers to write tests of their code into the original code. The result of the test statement should indicate that the code passed the test, or give some diagnostic information. Some thought has been given to implementation by macrogenerating acceptable code into an ALGOL program, or by modifying an ALGOL compiler to accept the metastatements.

3.4.4 Operating System

Full-time work on this part of the ILLIAC IV software started during this quarter. An overall approach to the problem was formulated and various design criteria defined. Operating system's personnel participated in

Work has begun on the mesh generator problem for the two dimensional heat equation for convex polygonal regions, with solution by means of alternating direction techniques. An algorithm has been developed and coded in ALGOL for simplified boundary conditions, and debugging on the B5500 has started.

4.1.2 User Support Codes

It was found, in attempting to analyze the problem by writing partial differential codes for ILLIAC IV, that a rather general structure might be used on a model for a large number of codes. (It would also serve for problems in areas other than partial differential equations.) This structure has five functionally distinct parts. In simple codes, some of these may be vestigial, but they can be identified. They are: 1) Data specification; 2) storage allocation; 3) input/output; 4) stencil operations; and 5) mathematical kernels.

The data specification includes mesh generation for PDE codes. In PDE problems, this segment may calculate coordinates of grid points, make lists of neighboring points, do interpolation for initial or boundary values, and set indices or flags to mark boundaries. Information supplied to this segment will include geometry data, mesh sizes, location of interfaces, and more.

Storage allocation assumes a crucial and unique role for ILLIAC IV. Efficient use of the machine depends directly on proper storage allocation. A number of canned mappings have been found to apply to a widening class of problems, and, as more of these somewhat universal allocation schemes are found, they will be added to the library. For some problems, it may be necessary to do a line by line or block by block allocation. Statements will be provided to do this.

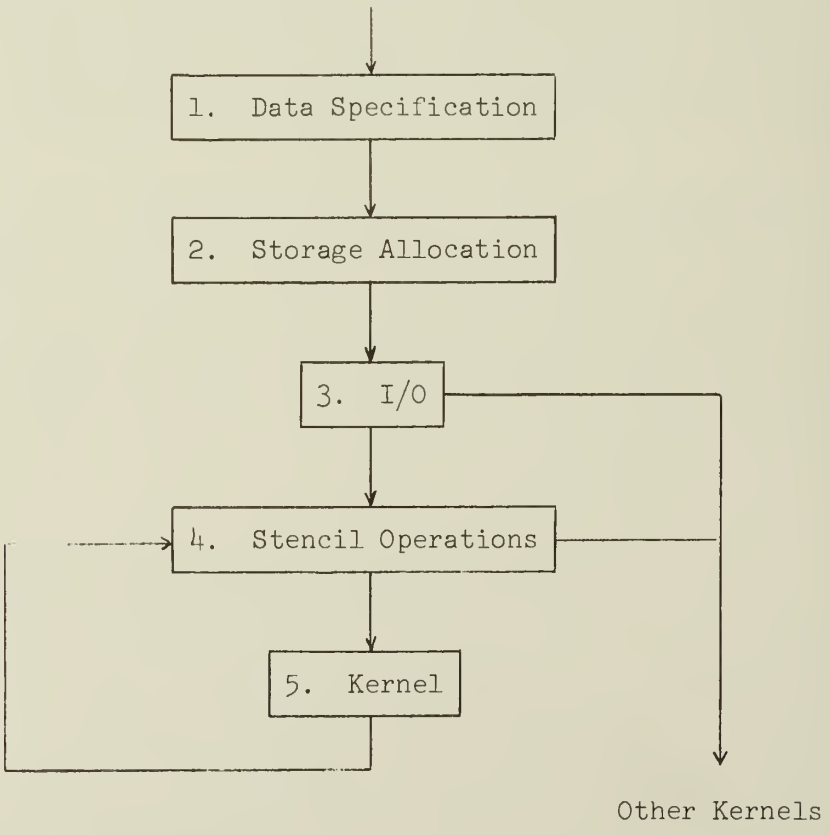
I/O, in addition to providing reading and writing of symbolic disk files and PE blocks, will also provide for graphical display of data through the B6500. In the latter area, it is expected to allow for graphing functions of two variables and simulated three-dimensional plots, with the ability to rotate and cross-section the image. It is also expected to provide for contour mapping and for the display of the map as it would appear on an arbitrary surface--i.e., a weather map on a hemisphere. It is also hoped

to have the facility for simulated three-dimensional contour mappings and for rotating and cross-sectioning the image. The graphical display package would also permit vector field graphs on arbitrary surfaces.

Stencil operations have been found useful in programming problems for irregular meshes and for expanding sparse matrices. They allow data to be addressed indirectly and they make resulting algorithms more regular and easier to write. On simple parts of the data structure, they may not be necessary.

The mathematical kernel is difficult to define exactly. In present thinking, it has come to be associated with a code without I/O which operates on stencils--the part of the code which is involved with setting up and solving the difference equation in a PDE program, inverting a matrix, or performing a sequence of transformation in an eigenvalue problem. Kernels may be written specifically or automatically generated from codes written for size independent problems.

Schematically, the following flow diagram pertains to this:



In an actual code, these segments interact in several ways. The compiler uses information from some parts in translating others--storage allocation is a declaration. At execution, parameters must be transmitted from one segment to another. Also, in real algorithms, many kernels will be linked together.

4.1.3 Slip Flow and Boundary Layer Studies

In any hydrodynamic problem involving the motion of a body in a fluid, the interest is in determining the velocity vector, the pressure, the temperature, etc., at any point in the flow field. The aerodynamic characteristics (lift, drag, etc.) of the body can then be obtained.

It is well established that the flow field can be divided into two regions: a very thin layer in the neighborhood of the body (boundary layer) where friction (viscous action) is important, and the remaining region outside this layer where viscous effects can be neglected. The governing equations for both regions--a system of nonlinear partial differential equations--are derived from the same principles which are the conservation of mass, momentum, and energy. The equations for the outer region, however, do not contain terms to account for viscous action and are called Euler's equations. For the boundary layer region, the resulting equations are the Navier-Stokes equations which are more complex.

Because of the great mathematical difficulties connected with solving the full Navier-Stokes equations, certain approximations are usually introduced into these equations to simplify them. Such approximations lead to the, so-called, boundary layer equations. The task of solving the boundary layer equations is the main and only theme of the boundary layer theory. It should be mentioned that the form of the boundary layer equations and the difficulty encountered in solving them depends on the geometry of the body and the manner in which it is moving.

The classical boundary conditions for the velocity vector W and the temperature are $W = 0$, $T = T_w$ respectively, at the surface of the body, where T_w is the wall temperature of the body. It has been established that in certain situations, these classical boundary conditions are not appropriate (the criterion is the value of a parameter called the rarefaction parameter).

In such circumstances, which are often encountered in outer space where the density is very low or in hypersonic speeds, the above boundary conditions are replaced by the velocity slip and temperature jump boundary conditions at the wall.

The simplest problem in which velocity slip rather than zero velocity should be used as a boundary condition is the slow motion of a flat plate in a rarefied flow field. The governing equations for this problem are:

$$U \frac{\partial U}{\partial x} + \frac{\partial U}{\partial y} = \nu \frac{\partial^2 U}{\partial y^2} \quad \nu = \text{constant},$$

$$\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} = 0$$

where all quantities are nondimensionalized by normalizing them. This system of equations can be reduced to a single second order partial differential equation by introducing the stream function Ψ and using the von Mises transformation. The resulting equation is:

$$\frac{\partial U}{\partial x} = \frac{1}{Re_{\gamma\infty}} \frac{\partial}{\partial \Psi} \left[U \frac{\partial U}{\partial \Psi} \right] \quad (1), \; U = U(\Psi, x)$$

where $U = \frac{\partial \Psi}{\partial y}$; $V = - \frac{\partial \Psi}{\partial x}$; $Re_{\gamma\infty}$ is the Reynold's number based on the free stream mean free path.

The transformed boundary conditions are:

$$U(\Psi, 0) = 1$$

$$\left.\frac{\partial U}{\partial \Psi}\right|(0, x) = 1$$

$$U(\infty, x) = 1$$

In the present study, equation (1) has already been solved by using a difference scheme suggested very recently by Milton Lees for a similar equation but with different boundary conditions than the above ones. The results of the solution were in good agreement with another solution developed in a laborious way by W. L. Chow who did not transform the system of equations to equation (1) and did not employ the mesh or grid technique to arrive at his numerical solution.

The success of the difference scheme used for equation (1) encouraged its application to a more sophisticated boundary layer problem in which the speed does not have to be slow, the density does not have to be low, and the heat transfer effects are not neglected. The governing equations for this suggested problem, in the X - Ψ space are:

$$\frac{\partial U}{\partial x} = \frac{1}{Re_{\gamma\infty}} \frac{\partial}{\partial \Psi} \left[T^{n-1} U \frac{\partial U}{\partial \Psi} \right],$$

$$\frac{\partial T}{\partial x} = \frac{Re_{\gamma\infty}}{Pr} \frac{\partial}{\partial \Psi} \left[U \frac{\partial T}{\partial \Psi} \right] + (\lambda - 1) M_{\infty}^2 U \left(\frac{\partial U}{\partial \Psi} \right)^2$$

where $U = U(\Psi, x)$ and $T = T(\Psi, x)$.

An attempt is now being made to solve the above system of equations, by using the velocity slip and temperature jump boundary conditions one time and by using the no slip, no jump boundary conditions the next time.

4.1.4 Matrix Operations

An algorithm for multiplying a sparse matrix times a sparse matrix was developed and coded in ILLIAC IV assembly language. To form the product A x B where A and B are sparse, the algorithm simultaneously forms dot products of the form $a_r \, b_i$ $1 \leq i \leq 256$ where a_r is a row vector of A, and b_i is a column vector of B [or a section of a row (column) vector of length 256 when A (B) is m x n, m > 256 (n > 256)].

The algorithm consists of two sections. The first section determines which multiplications have to be performed. Since these multiplications, in general, vary from column to column and thus from PE to PE, a queue consisting

of the nonzero elements of a_r that are to be used together with a tag denoting which element of b_i is to be used is formed in each PE.

The second section performs the multiplications using the information and the operands stored in the queue. Thus, a PE only has to be disabled when its total number of multiplications is less than the total number of multiplications of another PE.

To illustrate, consider a_r x B where:

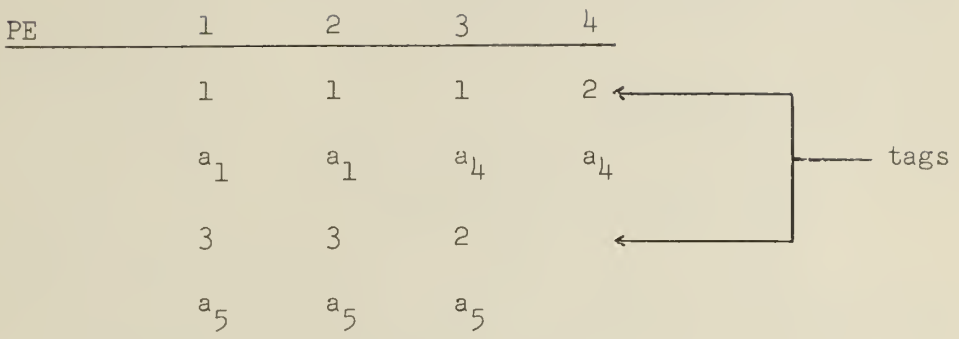
$$a_r = [a_1, 0, 0, a_4, a_5, 0]$$

$$B = \begin{bmatrix} b_{11} & b_{12} & 0 & 0 \\ 0 & b_{22} & 0 & b_{24} \\ b_{31} & 0 & 0 & 0 \\ 0 & 0 & b_{43} & b_{44} \\ b_{51} & b_{52} & b_{53} & 0 \\ 0 & b_{62} & 0 & b_{64} \end{bmatrix}$$

B is stored as:

PE	1	2	3	4
	b_{11}	b_{12}	b_{43}	b_{24}
	b_{31}	b_{22}	b_{53}	b_{44}
	b_{51}	b_{52}		b_{64}
		b_{62}		

The queue looks like:



Note that the tags do not refer to the row index of the b element but to the position of the element in the packed array.

4.1.5 Mathematical Service Routines

This quarter has seen the initiation of an effort to develop codes for standard algebraic and trigonometric functions and routines for ILLIAC IV. Several codes are being written, and the analysis for others has been undertaken.

A square root code (64 bit) has been completed. The technique used is essentially that of finding a close approximation and then applying Newton's method, a fixed number of times--a standard procedure.

A polynomial root finder using the techniques of multisectioning to find simple roots and then applying Bainstow's method for multiple roots is nearing completion.

Codes for log, sine, and cosine are under development.

4.1.6 Carlson's S_n Method

The programming of steady-state neutron transport in Tranquil was done during this quarter. The stead-state neutron transport was of the type existing in an homogeneous isotropic medium with no upscattering or independent source. Another characteristic was that it was of one dimensional spherical geometry assuming 32 velocity groups, 14 angular directions (Carlson's

discrete S_{14} formulation), and 128 spatial mesh points. At full efficiency, 8 processing elements calculate the inner solution for 32 groups simultaneously. Details are recorded in a separate ILLIAC IV document.

4.2 Signal Processing

4.2.1 Application of ILLIAC IV to Phased Array Radar for Area Defense

The use of the ILLIAC IV computer in handling data for large phased array radars for ICBM Area Defense has been investigated. A particular phased array radar system was modeled, assuming a certain number of objects, and programmed for single quadrant operation on ILLIAC IV. This single quadrant will be more than sufficient to handle the real time data processing requirements of radar control, scanning, designation, and tracking of potential targets.

The phased array radar model assumed has separate transmitter and receiver arrays. The radar system is capable of scanning 10^{11} resolution elements/second. These may have as many as 10^{8} apparent targets from which 3×10^{2} to 3×10^{3} should be tracked as possible targets for further discrimination. The transmitter has four quadrants which can operate, gather, or transmit four separate pulses simultaneously. Each quadrant has 12,544 radiating elements and requires 113 steering commands for each transmission. The receiver consists of clusters of elements where each cluster can operate independently. There are 20 clusters for scanning, 2 for designation, and 5 for tracking and multiple use; each cluster can receive 9 beams simultaneously.

The three major functions which require most of the computer time (steering commands, designation, track) have been programmed for ILLIAC IV operations and are presently undergoing simulation debugging for the B5500 computer. This major operation requires approximately 50% full computer capability of the single quadrant. A detailed report documenting the procedure for handling the problem is presently in the final stages of preparation.

4.3 Matrices

4.3.1 Eigenvalue Problems

In the last quarter, emphasis was given to eigenvalue problems of symmetric matrices. During this quarter, the eigenvalue problems for

non-symmetric matrices were dealt with (QR-algorithm). It was found that the QR-algorithm with a single origin shift will speed up the convergence greatly. It was also found that using a double origin shift is more advantageous in the case of complex-pair eigenvalues. The full report on Jacobi, Householder, and the QR-algorithm has been completed. The Tranquil language codes for the three methods are included in the report.

4.4 Linear Programming

4.4.1 Introduction

The study of linear programming has been directed to the writing of a totally, memory contained LP code in ILLIAC IV assembly language by using the general form revised simplex algorithm and to the investigation of a new algorithm for large scale linear programming problems.

4.4.2 LP Code

The totally contained LP code is quite suitable for the solution of relatively small scale linear programming problems, such as a size of 600 x 1000 or smaller. The main feature of this case is its simplicity, since no input-output operations are involved. A complete procedure description, including mathematical procedures and computational procedures, will soon appear in two ILLIAC IV Documents. This code will be simulated by the ILLIAC IV simulator using 64 bit precision. No other numerical provision is being included, such as reinversion of basis to restore the precision at some point of iteration.

The new algorithm for large scale linear programming problems attempts to make use of the great sparsity of the activity matrix and the formulation of the revised simplex method. This new algorithm is the product form revised simplex method which has been proven to be computationally effective in handling larger scale linear programming problems in which the general form revised simplex method is no longer effectively applicable. This new algorithm has received thorough investigation. Proper mathematical procedures have been formulated in terms of the product form revised simplex method and made applicable to the ILLIAC IV computer. A brief comparison between the general form revised simplex method and the product form revised simplex

method was made in order to visualize the advantages of the product form revised simplex method over the general form revised simplex method.

4.5 Computer Graphics

4.5.1 Introduction

Over the last quarter, ILLIAC IV's investigation of large scale weather calculations has reached the conclusion that general circulation modeling is a highly parallel process (as are most partial differential equations problems) to which ILLIAC IV can be efficiently applied. As in all applications, the real difficulty in using ILLIAC IV to simulate the large scale motion of the atmosphere lies in storing data across memory in such a way that the PE's can be used efficiently. In general, it seems that most global grid systems likely to be used in meteorology can be broken into large rectangle-like blocks, each of which can be stored in the machine in a straight-forward way. This, of course, does not eliminate the need for clever programming, but is a significant step towards parallelism.

4.5.2 NCAR Model

Two heartening results related to the NCAR model have appeared in the last quarter. The first involved a model comparable numerically to that coded and timed for ILLIAC IV, but not core contained. Also, assuming a disk block size of 256 words/PE, the ratio of block transmission time between disk and ILLIAC IV memory to computation time on 1 block of data is approximately 1 to 10. This allows I/O to be effectively masked by computation. The second involved a very simple calculation related to general circulation modeling. The ratio of execution time on NCAR's CDC 6600 (problem coded in FORTRAN IV) to estimated ILLIAC IV execution time (hand timed machine code) was 800 to 1--a substantial speedup.

4.5.3 Weather Mapping Algorithm

A great simplification occurred in the weather mapping algorithm when it was learned that the vectors do not need to be ordered to follow the contour lines. The reason for this is that weather maps are difficult to interpret if they contain too many contour lines. With a smaller number

of contour lines, a display is fast enough to draw the vectors in a random manner without flicker. It is now apparent that a complete weather map can be generated in about one millisecond witn all the work done in ILLIAC IV in a highly efficient manner.

4.5.4 Graphical Display System

Work this quarter has centered on debugging the graphical display system described in last quarter's QPR. The system is presently operating on an IBM 7094 with output to a Calcomp platter (via tape) and on a CDS 1604 with output direct tc a scope. Parts of the system have run on the B5500; but as that machine does not allow independent subroutines, there is doubt as to whether the present version of the system will be used on it. Present work is centered on detailed debugging and improvements of this system and on the definition and use of surfaces in this system.

4.6 Weapons Effects Calculations

During this period, the performance of the ILLIAC IV on weapons effects calculations was studied. The specific problems investigated were Eulerian hydrodynamics in two and three dimensions, and the solution of the radiation transport problem by the method of long characteristics.

Because of the assumed symmetry in the r direction in 2-D hydro, it is required that the number of cells in the Z direction equal roughly twice the number of cells in the r direction. Since 5 hydro quantities per cell must be stored, the total number of hydro cells must be less than 100,000. Subject to these constraints, the largest mesh that can be core contained with skewed storage is 256 x 125. However, there is a different method of storage allocation that will accommodate the entire 100,000 cells. The single material problem with a closed form equation of state was coded in assembly language and timed. It requires roughly 4000 clocks to advance the hydro quantities one time cycle in 256 cells. With an efficient I/O buffering scheme, it is possible to do very large mesh calculations with no apparent delays for the I/O.

No interesting 3-D problem can be core contained. Although the compute time per cell increases by about 30% over 2-D, the number of cells

that can be contained in a core load decreases by about a factor of 3. The I/O required per cell in 3-D is roughly double that required by 2-D. Because of these factors, the 3-D problem is I/O bound. If disc transfer rates were roughly doubled, the I/O bound would disappear.

The radiation transport problem was coded in Tranquil and then "hand" compiled. The hand compiled code (which may be quite different from the code produced by the compiler) requires about 5000 clocks to determine the radiation transport in a single direction and single frequency group across 256 eulerian cells. There appears to be no significant I/O problem.

Very interesting problems occur in multimaterial hydro (in the equation of state routine) and in the table look-up procedure required to do radiation transport. These problems are explained and possible solutions proposed in two ILLIAC IV documents now in preparation.

4.7 Anthropology Applications

4.7.1 Population Genetic Program

At this time, the experimentation with full chromosomal simulation in 'SCATRE' has been completed. This work demonstrated the feasibility of developing such a program for the ILLIAC IV. Although the model itself was shown to be operable, it is also apparent, both from the results of this experiment and from discussion with geneticists, that the scale and detail accuracy of the model would have to be improved for later exacting scientific use. A program is now under development in ALGOL for the B5500 which will test several other techniques for possible inclusion in an ILLIAC IV program. The original model was limited by small population size and small amounts of chromosomal material per individual, so that its development of trait forms was limited to a few small elementary linear, scalar, threshold characters with two levels of integration. The B5500 experiments will include several expansions. Instead of just simple phase shift and substitution mutations (which will also be done by a new algorithm), there will be included algorithms to simulate crossovers, loops, and other such higher level randomizing and sorting factors. A seven level non-scalar integrative and combinatorial plan for phenotype determination and several possible mating schemes will be tried along with some techniques of pseudo-simulation

(recording the state of the stable parts of the chromosome by a register indicating "changed/same"), thus saving space for more program density and population size. The code for several of the above algorithms has already been generated, but they have not yet been tested in a combined model. The exact form of the final ILLIAC IV model will depend on the results of these experiments.

4.7.2 Other Applications

Work on other applications has, to date, been confined to exploratory discussions of possible uses of computers in Anthropology with special respect being paid to the specific capabilities of the ILLIAC IV. Possible future programs include: Archeology--the establishment of n-dimensional matrices of time, space, and form related data of pot sherds and other artifacts, and the manipulation of these matrices to discover patterns of continuity both in strata and in evolutionary progression; Physical Anthropology--the application of transformation algorithms to constellations of points representing fossil material to determine continuities and aberations; Social Anthropology--the possibility has emerged that a fast efficient computer such as the ILLIAC IV will allow the expansion of some of the traditional theories of sociograms combined with the ideas of the social network theory and certain mathematical tools and concepts from graph theory such as the m-clique into a significant tool for ethnographic evaluation; Theoretical Anthropology--recent inquiries indicate a possible relationship between the ideas of Levi-Straus, the theory of binary cognitive models, and explorations in the character of such binary networks as are found in the retina of the eye; such that computer simulation might be a possible tool of analysis. Further applications are also under consideration.

REFERENCES

[1] Koga, Yoshiaki. Methods for Generating Diagnostic Sequences. (October 17, 1967), ILLIAC IV Document No. 160.

[2] Knowlton, Kenneth C. "A Programmer's Description of L^6." Comm. ACM 9, (August, 1966), pp. 616-625.

[3] Grothe, D. A Description of the Basic Assembly Language for ILLIAC IV, (October 5, 1967), ILLIAC IV Document No. 159.